The Event Calculus in Probabilistic Logic Programming with Annotated Disjunctions

Kevin McAreavey
Queen's University Belfast
Belfast, Northern Ireland
kevin.mcareavey@qub.ac.uk

Kim Bauters
University of Bristol
Bristol, England
kim.bauters@bristol.ac.uk

Weiru Liu
University of Bristol
Bristol, England
weiru.liu@bristol.ac.uk

Jun Hong
University of the West of England
Bristol, England
jun.hong@uwe.ac.uk

ABSTRACT

We propose a new probabilistic extension to the event calculus using the probabilistic logic programming (PLP) language ProbLog, and a language construct called the annotated disjunction. This is the first extension of the event calculus capable of handling numerous sources of uncertainty (e.g. from primitive event observations and from composite event definitions). It is also the first extension capable of handling multiple sources of event observations (e.g. in multi-sensor environments). We describe characteristics of this new extension (e.g. rationality of conclusions), and prove some important properties (e.g. validity in ProbLog). Our extension is directly implementable in ProbLog, and we successfully apply it to the problem of activity recognition under uncertainty in an event detection data set obtained from vision analytics of bus surveillance video.

Keywords

The event calculus, event reasoning, probabilistic logic programming, ProbLog, annotated disjunction

1. INTRODUCTION

The past decade has seen an increasing demand for so-called security information and event management (SIEM) platforms, such as IBM's QRadar.[1] SIEMs analyse security alerts generated by network hardware and applications, for the purpose of threat detection and prevention, decision support, and forensic analysis. Similar themes are reflected in vision-based activity recognition, e.g. for public transport security [16]. In either case, the aim is to support autonomous decision making and/or human decision support by exploiting large quantities of rich temporal data. Often this equates to inferring (composite) events of interest w.r.t. (primitive) event observations. Increasingly, the impact of uncertainty (e.g. in event observations, domain knowledge) is being recognised [36, 16]. Yet while SIEMs and activity recognition systems generally rely on rule-based components to reason about events, they are rarely founded on sound logical semantics. This is especially true of approaches capable of handling uncertainty (see [28] for a survey).

Reasoning about change (i.e. reasoning about events and their effects) is an important topic in AI. Some of the most influential logical approaches include the situation calculus [18], Allen's interval algebra [1], the event calculus [13] and action languages [9].[2] Conceptually, the event calculus is particularly suited for event reasoning orientated applications such as SIEM and activity recognition due to its use of a linear timeline of actual events as well as its ability to handle concurrent, incomplete and partially ordered events. Perhaps for this reason, the event calculus has continued to attract attention since its original proposal. However, the issue of uncertainty has been largely overlooked in the event calculus literature [19].

A notable exception is the work of Skarlatidis et al. In [30], they modelled a variant of the event calculus with discrete time using Markov logic networks (MLNs). This work handles uncertainty over the definitions of composite events but does not support uncertainty over event observations. Strong restrictions to the event calculus were also required to reduce the size of the ground MLN, resulting in the loss of canonical features, e.g. domain-independent capturing of events. In [29], they modelled a discrete-time variant of the event calculus in the probabilistic logic programming (PLP) language ProbLog [7]. This work (which is disjoint and incompatible with the previous one) handles some uncertainty over event observations but does not support uncertainty over the definition of composite events. Uncertainty over event observations also only relates to the occurrence of events (not to the time they occur, nor their properties).

In this paper, we propose a new extension to a continuous-time variant of the event calculus, using advances in PLP not considered by existing work. Our work provides the first formulation of the event calculus capable of handling numerous sources of uncertainty. As far as we are aware, it is also the first formulation capable of reasoning about events from multiple sources (e.g. in multi-sensor environ-

[1] http://www.ibm.com/software/products/qradar-siem

[2] Although many of these approaches are used for automated planning, it is not the subject of this paper.

Appears in: *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*

ments). Thus, it is particularly suited for event reasoning in SIEM and activity recognition systems. Our main contributions are: (i) we propose a new probabilistic definition for event observations, supporting uncertainty over their occurrence, the time at which they occur, and their associated properties; (ii) we propose new probabilistic definitions for effect clauses, initial beliefs, and composite events; (iii) we propose a method to handle multiple sources of event observations; (iv) we prove some important properties of our overall extension; and (v) we demonstrate an application of our extension in vision-based activity recognition.

The remainder of this paper is organised as follows: in Section 2 we recall some preliminaries; in Section 3 we propose our new extension; in Section 4 we examine properties of our extension; in Section 5 we summarise results from a case study on an event detection data set; in Section 6 we discuss related work; and in Section 7 we conclude.

2. PRELIMINARIES

In this section, we recall some preliminaries on logic programming (LP) and introduce an LP formulation of the event calculus. We also recall some preliminaries on the PLP language ProbLog which we will use to formalise our probabilistic extension of the event calculus in Section 3.

2.1 Logic programming

The language of logic programming (LP) is built from symbols for variables, constants, functors and predicates. By convention, variables start with uppercase letters and all others start with lowercase letters. If V is a variable and c is a constant, then V and c are terms. If f is a functor and $r_1, \ldots, r_n$ are terms s.t. $n \geq 1$, then $f(r_1, \ldots, r_n)$ is a term. If p is a predicate and $r_1, \ldots, r_n$ are terms s.t. $n \geq 0$, then $p(r_1, \ldots, r_n)$ is an atom. If p is an atom, then p and ***not*** p are (resp. positive and negative) literals. A (normal) clause is a formula $h :\!-\, b_1, \ldots, b_n$ s.t. h is an atom (called the head), $b_1, \ldots, b_n$ are literals (called the body), and all variables are implicitly universally quantified. Informally, if each b_i is true, then h must be true. If each b_i is an atom, then a clause is also called a definite clause. A clause $h :\!-\, true$ is called a fact and is simply written as h. An LP is a finite set of clauses. An expression is a term, atom, clause, or LP. Given an expression e and a substitution $\theta = \{V_1/r_1, \ldots, V_n/r_n\}$, then $e\theta$ denotes a new expression with each variable V_i in e replaced with the term r_i. Expressions are said to be ground if they contain no variables.

Herbrand models define the semantics of LP. The Herbrand base of an LP is the set of ground atoms w.r.t. the set of all possible ground terms (the Herbrand universe). A Herbrand interpretation is a mapping from the Herbrand base to the set of truth values $\{true, false\}$. For convenience, we will also refer a Herbrand interpretation ω by the set of atoms $\{p \in B \mid \omega(p) = true\}$ where B is the Herbrand base. A Herbrand interpretation is a Herbrand model of a definite clause $h :\!-\, b_1, \ldots, b_n$ if for every substitution θ s.t. $b_i\theta$ is a (set-theoretic) member of the interpretation, $h\theta$ is also a member of the interpretation. A Herbrand interpretation is a Herbrand model of a definite clause LP if it is a Herbrand model of every clause in the LP.

Definite clause LPs are guaranteed to have a unique minimal (by set inclusion) Herbrand model, called the least Herbrand model. A definite clause LP R is said to entail an atom p iff p is a member of the least Herbrand model of R. Thus, the least Herbrand model defines the semantics of definite clause LPs. Conversely, normal clause LPs are not guaranteed to have a least Herbrand model. For these programs, various alternative semantics have been proposed. As we will see later, ProbLog uses the well-founded semantics [33]. In this case, the (unique) two-valued well-founded model of an LP R is denoted $\mathsf{WFM}(R)$. An LP R is said to entail an atom p iff p is a member of $\mathsf{WFM}(R)$. Importantly, if R is a definite clause LP, then $\mathsf{WFM}(R)$ is identical to the least Herbrand model of R. However, $\mathsf{WFM}(R)$ is only guaranteed to exist for some classes of LP, such as those which are locally stratified [33].

Atom	Description
initially(F)	Fluent F holds at timepoint 0
happens(E, T)	Event type E occurs at timepoint T
initiates(E, F, T)	At timepoint T, event type E initiates a period during which fluent F holds
terminates(E, F, T)	At timepoint T, event type E terminates a period during which fluent F holds

Table 1: Domain-dependent atoms in the SEC.

2.2 The event calculus

The event calculus provides a mechanism to reason about temporal information in classical logic using the commonsense law of inertia, which says that properties (called fluents) persist over time unless the occurrence of an event triggers a change. As formalised in LP, the event calculus is a kind of many-sorted LP, with sorts for event types, fluents, and timepoints. A set of domain-independent axioms specify which fluents hold at any given timepoint w.r.t. a timeline (narrative) of event type occurrences and a set of domain-dependent effect clauses. The frame problem, i.e. the apparent need to explicitly model the non-effects of events, is addressed in the event calculus through the use of non-monotonic reasoning. First-order logic formulations rely on circumscription [17], while LP formulations rely on some semantics for negation by default (e.g. completion [6] or stable models [14]). The well-founded semantics is an example of the latter.

There is no definitive variant of the event calculus (see [27, 19] for surveys). Nonetheless, the simplified event calculus (SEC) from [27] arguably represents the intersection of most variants. The SEC is essentially a simplification of a revised variant [12] of the original event calculus [13]. Throughout this paper, we use E, F and T to denote variables for event types, fluents and timepoints, respectively. We also assume the existence of a partial order $\preceq$ over timepoints s.t. for each timepoint T we have that $0 \preceq T$. The SEC is then defined by the atoms in Table 1 and the following axioms:

$$\begin{aligned} &\mathit{holdsAt}(F, T_2) :\!- \\ &\quad \mathit{happens}(E, T_1), \\ &\quad \mathit{initiates}(E, F, T_1), \\ &\quad T_1 \prec T_2, \\ &\quad \textbf{\textit{not}}\ \mathit{stoppedIn}(T_1, F, T_2). \end{aligned} \tag{SEC1}$$

$$\begin{aligned} &\mathit{stoppedIn}(T_1, F, T_2) :- \\ &\quad \mathit{happens}(E, T), \\ &\quad \mathit{terminates}(E, F, T), \\ &\quad T_1 \prec T, T \prec T_2. \end{aligned} \tag{SEC2a}$$

$$\begin{aligned} &\mathit{clipped}(T_1, F, T_2) :- \\ &\quad \mathit{happens}(E, T), \\ &\quad \mathit{terminates}(E, F, T), \\ &\quad T_1 \preceq T, T \prec T_2. \end{aligned} \tag{SEC2b}$$

$$\begin{aligned} &\mathit{holdsAt}(F, T) :- \\ &\quad \mathit{initially}(F), \\ &\quad \textbf{\textit{not}}\ \mathit{clipped}(0, F, T). \end{aligned} \tag{SEC3}$$

In its original definition, the SEC was comprised of SEC1, SEC2a (called SEC2) and SEC3, with the second literal in SEC3 replaced with ***not*** *stoppedIn(0, F, T)*. However, such a definition means that if an event occurs at timepoint *0* and terminates a fluent *f* that holds initially, then that effect will be ignored since ***not*** *stoppedIn(0, f, t)* will hold for any timepoint $t \succeq 0$. To avoid this issue, we introduce SEC2b and modify the second literal in SEC3, as was done in later extensions [27, 19]. Notable features supported by the SEC include: (i) continuous time; (ii) concurrent events; (iii) partially ordered events; (iv) conditional and context-dependent effects; and (v) inferred (i.e. conditional, composite or hierarchical) events. Other features which can be supported via extensions to the SEC include: events with duration; cumulative and canceling effects; release of fluents from the commonsense law of inertia; and continuous change. Due to space considerations, we will focus on the SEC but our conclusions extend to more elaborate variants.

2.3 ProbLog

There are numerous PLP languages, each with subtle differences in syntax, semantics and implementation [20, 22, 25, 35, 34]. For continuity with related work [29], we will define our framework in the ProbLog language [7]. Nonetheless, it is likely that our framework will map to at least some other PLP languages, and we will suggest some possibilities in Section 7 when we mention future work.

A ProbLog program $F \cup R$ consists of a set of ground probabilistic facts F and an LP R (including non-probabilistic facts). Probabilistic facts are denoted $p :: a$ s.t. a is an atom (called a probabilistic atom) and p is a probability value. As with most PLPs, a ProbLog program specifies a probability distribution over Herbrand interpretations via Sato's distribution semantics [24]. Intuitively, a ground probabilistic fact $p :: a$ corresponds to an independent atomic choice about including a in a set of atoms C s.t. $C \cup R$ forms a new LP. In particular, $p :: a$ says that a should be included in C with probability p and not included with probability $1-p$. Thus, a set of probabilistic atoms C represents a total choice over the inclusion of ground atoms from F in $C \cup R$. Due to the independence assumption, the probability of a total choice C is equal to the product of the probabilities of atomic choices specified by C.

A Herbrand interpretation ω is a model of a ProbLog program $F \cup R$ iff there exists a total choice C s.t. $\mathsf{WFM}(C \cup R) = \omega$. A probability distribution P over the set of Herbrand interpretations Ω is defined for each $\omega \in \Omega$ as:

$$\mathsf{P}(\omega) = \begin{cases} \mathsf{P}(C), & \text{if } \omega = \mathsf{WFM}(C \cup R), \\ 0, & \text{otherwise.} \end{cases}$$

If there exists a total choice C s.t. $\mathsf{WFM}(C \cup R)$ does not exist, $F \cup R$ is not a valid ProbLog program. Otherwise, for any total choices C and C', it is guaranteed that $\mathsf{WFM}(C \cup R) \neq \mathsf{WFM}(C' \cup R)$. Thus, there exists a unique model for each total choice, but this model may have a probability of 0 (i.e. if the total choice itself has a probability of 0).

The definition of a ProbLog program may appear restrictive at first. However, for ease of modelling, ProbLog provides some additional syntactic sugar. One example is the annotated disjunction (AD) [7], which is a language construct of the form:

$$p_1 :: h_1; \ldots; p_n :: h_n :- b_1, \ldots, b_m$$

where $n \geq 1$ and $\sum_{i=1}^{n} p_i \leq 1$. Intuitively, an AD says that if each b_i is true, then exactly one h_i is true and this h_i is true with probability p_i. Conversely, every h_i is false with probability $1 - \sum_{i=1}^{n} p_i$. In a sense, the ground atoms $h_1, \ldots, h_n$ represent the possible values of a multi-valued random variable. In ProbLog, ADs are said to be syntactic sugar because they are encoded as a set of probabilistic facts and a set of (deterministic) normal clauses [35, 7].

Example 1. Consider the AD $\frac{1}{3} :: \mathit{red}$; $\frac{1}{3} :: \mathit{green}$; $\frac{1}{3} :: \mathit{blue}$. This is encoded as the following ProbLog program [7]:

$$\begin{gathered} \tfrac{1}{3} :: \mathit{choice(g)}. \qquad \tfrac{1}{2} :: \mathit{choice(r)}. \qquad 1 :: \mathit{choice(b)}. \\ \mathit{green} :- \mathit{choice(g)}. \qquad \mathit{red} :- \textbf{\textit{not}}\ \mathit{choice(g)}, \mathit{choice(r)}. \\ \mathit{blue} :- \textbf{\textit{not}}\ \mathit{choice(g)}, \textbf{\textit{not}}\ \mathit{choice(r)}, \mathit{choice(b)}. \end{gathered}$$

In this encoding, independence between probabilistic facts is still assumed, but dependencies are captured by normal clauses after a suitable adaption of probability values. Consider the following clause:

$$\mathit{conjunction} :- \mathit{green}, \mathit{red}, \mathit{blue}.$$

Then $\mathsf{P}(\mathit{conjunction}) = 0$. Conversely, suppose the original ProbLog program only contained the independent probabilistic facts $\frac{1}{3} :: \mathit{red}$, $\frac{1}{3} :: \mathit{green}$ and $\frac{1}{3} :: \mathit{blue}$, then we would have $\mathsf{P}(\mathit{conjunction}) = \frac{1}{27}$. □

Different implementations of ProbLog use different methods to evaluate the probability distribution over Herbrand interpretations. This paper is agnostic to the particular method used and instead focuses on features of the language that are useful to adapt the event calculus into PLP. However, as only ProbLog2 [8] supports ADs, our discussion will assume the use of Problog2 in practice.

3. A PROBLOG EXTENSION OF THE SEC

In this section, we propose a ProbLog extension of the simplified event calculus (SEC) and describe its characteristics. The main aspects that we need to consider are: (i) event narratives; (ii) event effects; (iii) initial beliefs; (iv) inference rules; and (v) multiple sources of events.

3.1 Event narrative

In the SEC, the event narrative is a set of ground atoms of the form $\mathit{happens}(e, t)$. This was extended in [29] as a set of ground probabilistic facts of the form $p :: \mathit{happens}(e, t)$

so as to model uncertainty over the occurrence of events. However, given that probabilistic facts are independent, this approach is not capable of modelling uncertainty over the time at which an event occurs. We propose the following:

Definition 1. *Let e be an event type and $t_1, \ldots, t_n$ be timepoints. A probabilistic occurrence of e is a ground annotated disjunction (AD):*

$$p_1 :: happens(e, t_1); \ldots; p_n :: happens(e, t_n).$$

This says that event type e occurs at timepoint t_i (and not at timepoint t_j s.t. $t_j \neq t_i$) with probability p_i and does not occur (at any timepoint $t_1, \ldots, t_n$) with probability $1 - \sum_{i=1}^{n} p_i$. Clearly, this form of uncertainty cannot be expressed purely by probabilistic *happens(E, T)* facts since they cannot enforce mutual exclusion between occurrences.

Another form of uncertainty in the event narrative relates to event type definitions. In particular, the term e in the atom $happens(e, t)$ is generally assumed to provide sufficient information to reason about the effects of an event. However, as was noted in the original proposal of the event calculus [13], event type definitions may be incomplete. The solution proposed in their work is to refer to an event type by an identifier and to describe it by a set of properties (attributes). Effectively, these attributes construct a semantic network where edges are specified by ground atoms of the form *h(e, r)* s.t. r is the value of attribute h associated with event type e. Given that event type definitions describe the actual meaning of events, they must be capable of modelling uncertainty. We propose the following:

Definition 2. *Let h be a binary predicate symbol, e an event type, and $r_1, \ldots, r_n$ be terms. A probabilistic h attribute for e is a ground AD:*

$$p_1 :: h(e, r_1); \ldots; p_n :: h(e, r_n).$$

Clearly Definition 2 generalises Definition 1, saying that the value of attribute h for event type e is r_i with probability p_i and is unknown with probability $1-\sum_{i=1}^{n} p_i$. Thus, p_i can be thought of as a degree of belief that the value of h is r_i, where $1-\sum_{i=1}^{n} p_i$ is a degree of ignorance over the value of h. Once again, probabilistic *h(E, R)* facts cannot express this notion since they cannot enforce mutual exclusion over the possible values of h. Notice that we do not allow ADs of the form $p_1 :: happens(e_1, t_1); \ldots; p_n :: happens(e_n, t_n)$ in Definition 1. The reason being that $e_1, \ldots, e_n$ can be equivalently expressed by a single event type e and a set of probabilistic attributes for e. Moreover, relying on a single event type simplifies other aspects of this extension, e.g. when merging event narratives.

In our extension, a set of probabilistic event type occurrences and a set of probabilistic event type attributes is called a probabilistic event narrative. Given an event type e, we refer to the set of probabilistic attributes for e and a single probabilistic occurrence of e as a probabilistic event. Thus, for any probabilistic event, we are capable of expressing uncertainty about whether the event actually occurred, as in [29], but *also* about the time at which it occurred, and about its attributes. In ProbLog, a probabilistic event narrative actually specifies a probability distribution over the set of total choices, corresponding to a set of (classical) event narratives. If any such narrative contains more than one value specified by a probabilistic event type attribute, then that narrative will have a probability of 0.

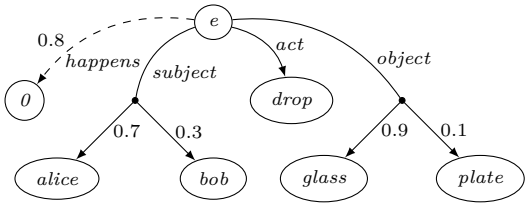

Figure 1: Semantic network for Example 2.

Example 2. Consider an event where one of two people (Alice with probability 0.7, Bob with probability 0.3) is thought to have dropped either a glass (with probability 0.9) or a plate (with probability 0.1). The event is believed to have occurred with probability 0.8. This could be modelled by the following probabilistic event:

$$0.8 :: happens(e, 0).$$
$$0.7 :: subject(e, alice); 0.3 :: subject(e, bob).$$
$$act(e, drop).$$
$$0.9 :: object(e, glass); 0.1 :: object(e, plate).$$

In ProbLog, this induces the following probability distribution over total choices (we omit e and predicate symbols):

$$\mathsf{P}(\{0, alice, glass\}) = 0.504, \quad \mathsf{P}(\{alice, glass\}) = 0.126,$$
$$\mathsf{P}(\{0, alice, plate\}) = 0.056, \quad \mathsf{P}(\{alice, plate\}) = 0.014,$$
$$\mathsf{P}(\{0, bob, glass\}) = 0.216, \quad \mathsf{P}(\{bob, glass\}) = 0.054,$$
$$\mathsf{P}(\{0, bob, plate\}) = 0.024, \quad \mathsf{P}(\{bob, plate\}) = 0.006,$$

with $\mathsf{P}(C) = 0$ for any other total choice C. Thus we can infer from this probabilistic event e.g. that the glass was dropped at timepoint 0 with probability 0.72. □

3.2 Effect clauses

In the SEC, events are not of direct interest. Rather, their interest lies in their effect on fluents w.r.t. domain-dependent effect clauses with *initiates(E, F, T)* and *terminates(E, F, T)* as the head. The event model outlined previously has interesting consequences for such effect clauses and the fluents that they initiate or terminate. Specifically, if p is the probability of models specified by a probabilistic event (e.g. 0.72 in Example 2) and satisfying an effect clause, then this will propagate to the specified fluent, causing it to be initiated or terminated with probability p. Thus, each fluent will hold with some probability. This behaviour was recognised in [29], where events had the effect of reinforcing previously held beliefs (i.e. increasing or decreasing previously held probabilities). However, our approach is necessarily different due to the use of ADs. Specifically, given some domain of a probability distribution $f_1, \ldots, f_n$, then for any ground $initiates(e, f_i, t)$ clause there must also be ground $terminates(e, f_1, t), \ldots, terminates(e, f_n, t)$ clauses. In effect, this approach maintains a valid probability distribution over $f_1, \ldots, f_n$ by terminating the previously held probabilities each time that new probabilities are initialised. This is akin to database updates as described in [12] and is imposed as a syntactic restriction.

Example 3. Consider Example 4 from [15], which has six belief update events describing the gender and/or orientation of a subject over six timepoints. For example, the first event says that subject s_1 is standing, and that their gender

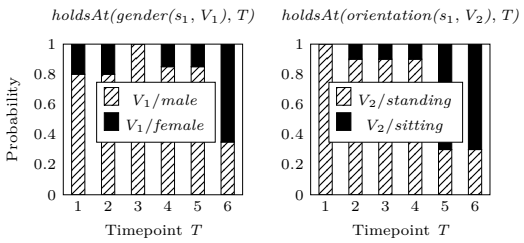

Figure 2: Probabilities of atoms in Example 3.

is male with probability 0.8 or female with probability 0.2. These can be modelled by the following probabilistic events:

$happens(e_1, 0).\ subject(e_1, s_1).\ orientation(e_1, standing).$
$0.8 :: gender(e_1, male); 0.2 :: gender(e_1, female).$

$happens(e_2, 1).\ subject(e_2, s_1).$
$0.9 :: orientation(e_2, standing); 0.1 :: orientation(e_2, sitting).$

$happens(e_3, 2).\ subject(e_3, s_1).\ gender(e_3, male).$

$happens(e_4, 3).\ subject(e_4, s_1).$
$0.85 :: gender(e_4, male); 0.15 :: gender(e_4, female).$

$happens(e_5, 4).\ subject(e_5, s_1).$
$0.3 :: orientation(e_5, standing); 0.7 :: orientation(e_5, sitting).$

$happens(e_6, 5).\ subject(e_6, s_1).$
$0.35 :: gender(e_6, male); 0.65 :: gender(e_6, female).$

Suppose we have the following effect clauses:

$$initiates(E, gender(S, G), T) :- subject(E, S), gender(E, G).$$
$$terminates(E, gender(S, G_1), T) :- subject(E, S), gender(E, G_2).$$

with equivalent clauses defined for $orientation(S, G)$.[3] Then the resulting *holdsAt* probabilities are as outlined in Figure 2, matching the desired behaviour in [15]. □

This example demonstrates that our probabilistic event model allows us to derive rational probabilistic conclusions via standard (deterministic) effect clauses. However, it is commonly accepted in the literature (e.g. in planning under uncertainty) that the effects of events may not be deterministic. Thus, there is a clear need to support the definition of such effects in the SEC. We propose:

Definition 3. *Let E be an event type, $F_1, \ldots, F_n$ be fluents and $T_1, \ldots, T_n$ be timepoints. A probabilistic effect clause for E is an AD:*

$$p_1 :: h(E, F_1, T_1); \ldots; p_n :: h(E, F_n, T_n) :- b_1, \ldots, b_m$$

where $h \in \{initiates, terminates\}$.

This says that, if each b_i is true, then with probability p_i event type E initiates (resp. terminates) a period at timepoint T_i during which fluent F_i holds. Conversely, with probability $1 - \sum_{i=1}^{n} p_i$, event type E does not initiate (resp. terminate) this period. As with deterministic effect clauses, the event probabilities are propagated to conclusions. In this case, however, conclusions themselves are also uncertain.

Example 4 (Continuing Example 2). Suppose that, when dropped, the glass may break with probability 0.9 and the plate may break with probability 0.8. This can be represented by the following probabilistic effect clauses:

$$0.9 :: initiates(E, broken(glass), T) :- act(E, drop), object(E, glass).$$
$$0.8 :: initiates(E, broken(plate), T) :- act(E, drop), object(E, plate).$$
$$terminates(E, broken(O), T) :- act(E, drop), object(E, O).$$

Thus, for any timepoint $t \succ 0$, we can infer that the glass is broken with probability 0.648, and the plate is broken with probability 0.064. □

3.3 Initial beliefs

Aside from effect clauses, the SEC also allows us to specify which fluents hold prior to the occurrence of events by means of a set of ground atoms of the form *initially(f)*, as indicated in Table 1. We can propose a simple probabilistic extension:

Definition 4. *Let $f_1, \ldots, f_n$ be fluents. Then an initial probabilistic belief is a ground AD:*

$$p_1 :: initially(f_1); \ldots; p_n :: initially(f_n).$$

This says that fluent f_i holds with probability p_i and that fluents $f_1, \ldots, f_n$ do not hold with probability $1 - \sum_{i=1}^{n} p_i$. In practice, initial probabilistic beliefs define the initial context for events (e.g. for determining conditional effects) and persist over time until an event occurs with effects over the domain $f_1, \ldots, f_n$. From this point, probabilities over $f_1, \ldots, f_n$ will be determined by events and effect clauses. Note that initial probabilistic beliefs do not impose any requirement on complete knowledge. For example, omitting initial beliefs that are unknown allows the SEC axioms to derive (or not derive) conclusions as usual, while incomplete initial probabilistic information is expressed when $\sum_{i=1}^{n} p_i < 1$.

Example 5 (Continuing Example 4). Suppose instead that the probability of the glass breaking is dependent on the person's orientation when they drop the glass (probability 0.9 if standing and 0.3 if sitting). This can be represented by the following probabilistic effect clauses:

$$0.9 :: initiates(E, broken(glass), T) :- subject(E, S), act(E, drop), object(E, glass), holdsAt(orientation(S, standing), T).$$
$$0.3 :: initiates(E, broken(glass), T) :- subject(E, S), act(E, drop), object(E, glass), holdsAt(orientation(S, sitting), T).$$

Suppose our prior belief is that Alice is standing with probability 0.6 and sitting with probability 0.4, and that Bob is standing (resp. sitting) with probability 0.5. This can be modelled by the following initial probabilistic beliefs:

$0.6 :: initially(orientation(alice, standing))$;
$0.4 :: initially(orientation(alice, sitting))$.

[3] While some properties (e.g. gender) are not fluents in principle, what we are describing is *beliefs* about those properties and this satisfies the intuition of a fluent.

$0.5 :: initially(orientation(bob, standing))$;
$0.5 :: initially(orientation(bob, sitting))$.

Thus, for any timepoint $t \succ 0$, we can infer that the glass is broken with probability 0.462. □

3.4 Inference rules

Composite events are closely related to other concepts, such as conditional events. In [16], they refer to any event which is derived through reasoning as an inferred event, i.e. conditional, composite, and hierarchical events are all special cases. We will use this term also. In the SEC, inferred events can be defined by means of inference rules. As with effect clauses, such rules may be uncertain. We propose:

Definition 5. *Let E be an event type, $T_1, \ldots, T_x$ timepoints, and $h_1(E, V_1), \ldots, h_y(E, V_y)$ event type attributes for E. A probabilistic inference rule for E is an AD:*

$p_1 :: infer(happens(E, T_1), h_1(E, V_1), \ldots, h_y(E, V_y)); \ldots;$
$p_x :: infer(happens(E, T_x), h_1(E, V_1), \ldots, h_y(E, V_y)) :-$
$b_1, \ldots, b_z$.

Ground terms in the head of a probabilistic inference rule can be interpreted as the definition of a new probabilistic event (called a probabilistic inferred event) of the form:

$p_1 :: happens(E, T_1); \ldots; p_x :: happens(E, T_x)$.
$h_1(E, V_1). \ \ldots \ h_y(E, V_y)$.

A probabilistic inference rule is able to derive a new event with uncertainty over its occurrence and over the time at which it occurred, but not over its attributes. However, to ensure that our extension satisfies the well-founded semantics (as will be explained in Section 4), we do not support hierarchical events, i.e. inferred events derived from other inferred events. In this sense, inferred events are an output of our extension. That said, it is possible to support a kind of hierarchical event by inserting inferred events into the event narrative as new (primitive) event observations, although this requires a revision strategy to maintain the set of inferred events.

Example 6 (Continuing Example 3). Suppose we want to recognise an event in which a subject sits down. Given that we have temporal beliefs about the subject's orientations, we can infer such an event via changes from the standing to sitting orientations. Assume this recognition is accurate with probability 0.9. This can be modelled by the following probabilistic inference rule:

$0.9 :: infer(happens(e(S, sit), T),$
$subject(e(S, sit), S), act(e(S, sit), sit)) :-$
$happens(E, T), \ subject(E, S), \ orientation(E, sitting),$
$holdsAt(orientation(S, standing), T)$.

The resulting (inferred) event narrative is as follows:

$0.09 :: happens(e(s_1, sit), 1)$. $\quad 0.567 :: happens(e(s_1, sit), 4)$.
$subject(e(s_1, sit), s_1)$. $act(e(s_1, sit), sit)$.

Thus, subject s_1 sits down at timepoint *1* with probability 0.09, and at timepoint *4* with probability 0.567. □

In [15], a similar scenario was suggested where sudden changes to the gender assessment could be used as an indication of attempts to obfuscate the gender classifier. However, the authors did not provide a formal mechanism to detect such an event. Example 6, on the other hand, demonstrates that our framework is capable of implementing this feature.

$subject(e, X)\theta$	P_1	P_2	P_3	$\oplus(\mathsf{P}_1, \mathsf{P}_2, \mathsf{P}_3)$
$\theta = \{X/alice\}$	0.7	0.8	0.4	0.67
$\theta = \{X/bob\}$	0.3	0	0.1	0.17
$\theta = null$	0	0.2	0.5	0.16

Table 2: Linear opinion pool for Example 7.

3.5 Merging event narratives

In real-world applications, event reasoning systems usually operate in multi-sensor environments, e.g. SIEMs aggregate events obtained separately from log files and network traffic. Clearly, different sources may provide conflicting observations of the same event. As implied previously, a probabilistic event type attribute $p_1 :: h(e, r_1); \ldots; p_n :: h(e, r_n)$ expresses a probability distribution P over the set of values $\Omega = \{r_1, \ldots, r_n, unknown\}$ for h s.t. $\mathsf{P}(r_i) = p_i$ and $\mathsf{P}(unknown) = 1 - \sum_{i=1}^{n} p_i$. In this sense, the problem of merging event narratives reduces to the problem of merging (aggregating) subjective probabilities. A standard approach to this problem is the linear opinion pool [31], which is a linear weighted average defined for each $\omega \in \Omega$ as follows:

$$\oplus(\mathsf{P}_1, \ldots, \mathsf{P}_n)(\omega) = \sum_{i=1}^{n} w_i \mathsf{P}_i(\omega)$$

where w_i is a weight associated with source s_i s.t. $0 \leq w_i \leq 1$ and $\sum_{i=1}^{n} w_i = 1$. When there is no confusion, we will use P to denote a probabilistic h attribute for e.

Definition 6. *Let $\mathsf{P}_1, \ldots, \mathsf{P}_n$ be probabilistic h attributes for e obtained from sources $s_1, \ldots, s_n$. Then the merged probabilistic h attribute for e is defined as $\oplus(\mathsf{P}_1, \ldots, \mathsf{P}_n)$.*

Example 7. Consider the probabilistic *subject* attributes for e provided by sources s_1, s_2 and s_3 as outlined in Table 2 s.t. $w_1 = 0.5$, $w_2 = 0.3$ and $w_3 = 0.2$. We obtain the following merged probabilistic *subject* attribute for e:

$0.67 :: subject(e, alice); 0.17 :: subject(e, bob)$. □

Suppose there exists a unique event type for each real-world event, the event narrative from each source is defined over the same set of event types, and each event narrative has the same probabilisitic h attributes for each event type (e.g. assume $\mathsf{P}(unknown) = 1$ for missing attributes). We can thus obtain a merged event narrative by merging each probabilistic h attribute w.r.t. each event type e. Note that the choice of the linear opinion pool is merely to illustrate the principle that combining event narratives reduces to the problem of combining subjective probabilities, i.e. any other valid merging operator could be used instead.

4. PROPERTIES

We now examine properties of our new extension. Firstly, we define an SEC ProbLog program as a ProbLog program $N \cup E \cup I \cup C \cup \{\text{SEC1}, \ldots, \text{SEC3}\}$ where N is a probabilistic event narrative, E is a set of probabilistic effect clauses, I is a set of initial probabilistic beliefs, and C is a set of probabilistic inference rules. The clauses with p as the head

are called the definition of p and are denoted simply by p itself. We must clarify some assumptions: **A1** SEC1–3 are the exhaustive definitions of *holdsAt(F, T)*, *stoppedIn(*T_1, *F*, T_2*)* and *clipped(*T_1, *F*, T_2*)*; **A2** with the exceptions of SEC1 and SEC3, *stoppedIn(*T_1, *F*, T_2*)* and *clipped(*T_1, *F*, T_2*)* do not occur as positive/negative literals in the body of any clause; and **A3** *infer(*$H_1, \ldots, H_n$*)* does not occur as a positive/negative literal in the body of any clause. Without loss of generality, we assume all ADs are subsequently translated to probabilistic facts and normal clauses.

Proposition 1. *Let $F \cup R$ be an SEC ProbLog program in which bodies of clauses in the definitions of initiates(E, F, T) and terminates(E, F, T) do not contain literals of the form holdsAt(F, T),* ***not*** *holdsAt(F, T),* ***not*** *initiates(E, F, T) or* ***not*** *terminates(E, F, T). Then, for each total choice C, we have that $C \cup R$ is a stratified LP.*

Proof. An LP R is stratified if there exists a partition $R = R_0 \cup \cdots \cup R_n$, s.t. $R_i \cap R_j = \emptyset$ for any $i \neq j$, satisfying: (i) for each atom p, the definition of p is a subset of some R_i; (ii) for each $1 \leq i \leq n$, if a literal p is in the body of a clause in R_i, then the definition of p is a subset of some R_j s.t. $j \leq i$; and (iii) for each $1 \leq i \leq n$, if a literal ***not*** p is in the body of a clause in R_i, then the definition of p is a subset of some R_j s.t. $j < i$. Let $\{p_1 :: c_1(H), \ldots, p_x :: c_x(H)\} \subseteq F$ be new probabilistic facts after translation of ADs. Given assumptions **A1**–**3** and the fact that probabilistic event type attributes have empty bodies by Definition 2, then the following is a valid stratification of $C \cup R$:

$$
\begin{aligned}
(C \cup R)_0 &= \{\preceq, c_1(H), \ldots, c_x(H)\},\\
(C \cup R)_1 &= \{\mathit{initially(F)}, \mathit{happens(E, T)},\\
&\qquad h_1(E, V), \ldots, h_y(E, V)\},\\
(C \cup R)_2 &= \{\mathit{initiates(E, F, T)}, \mathit{terminates(E, F, T)},\\
&\qquad \mathit{stoppedIn(}T_1, F, T_2\mathit{)}, \mathit{clipped(}T_1, F, T_2\mathit{)}\},\\
(C \cup R)_3 &= \{\mathit{holdsAt(F, T)}\},\\
(C \cup R)_4 &= \{\mathit{infer(}H_1, \ldots, H_z\mathit{)}\}.
\end{aligned}
$$

Therefore, $C \cup R$ is a stratified LP. □

Corollary 1. *Let $F \cup R$ be an SEC ProbLog program satisfying Proposition 1. Then $F \cup R$ is a valid ProbLog program.*

The restrictions imposed in Corollary 1 are somewhat limiting in practice. For example, literals *holdsAt(F, T)* and ***not*** *holdsAt(F, T)* are prohibited, even though they are required to define context-dependent effects. Fortunately, these restrictions are stronger than is actually necessary, since we only need to ensure that an SEC ProbLog program is locally stratified.

Proposition 2. *Let $F \cup R$ be an SEC ProbLog program which does not contain looping through ground definitions of initiates(e, f, t), terminates(e, f, t) and holdsAt(f, t). Then, for each total choice C, we have that $C \cup R$ is a locally stratified LP.*

Proof. Given Proposition 1, it remains to be shown that $(C \cup R)_2 \cup (C \cup R)_3$ is locally stratified. An LP R is said to be locally stratified if there exists a grounding $R\theta$ s.t. $R\theta$ is stratified. Let the ground definition of atom *holdsAt(*f_1, t_1*)* be in strata k. Then the ground definitions of *stoppedIn(*e_1, f_1, t_1*)* and *clipped(*e_1, f_1, t_1*)* can be in some strata $j < k$. Moreover, the ground definitions of atoms *initiates(*e_1, f_1, t_1*)* and *terminates(*e_1, f_1, t_1*)* can be in some strata $i < j$. Let the body of some clause in the ground definitions of either atom *initiates(*e_1, f_1, t_1*)* or *terminates(*e_1, f_1, t_1*)* contain a positive/negative literal of the form *holdsAt(*f_2, t_2*)*, *initiates(*e_2, f_2, t_2*)* or *terminates(*e_2, f_2, t_2*)*. Then the ground definition of atom *holdsAt(*f_2, t_2*)* can be in some strata $h < i$. If there is no looping through ground definitions of *initiates(*e_1, f_1, t_1*)*, *terminates(*e_1, f_1, t_1*)* and *holdsAt(*f_1, t_1*)*, then there exists some total stratification in this manner. Therefore, $C \cup R$ is a locally stratified LP. □

Corollary 2. *Let $F \cup R$ be an SEC ProbLog program satisfying Proposition 2. Then $F \cup R$ is a valid ProbLog program.*

This proves that the semantics of our extension to the SEC are indeed well-defined under natural conditions, i.e. we only require that an SEC ProbLog program does not contain (infinite) looping through effect clauses.

5. CASE STUDY

The aim of [11] was to augment vision analytics with composite event detection for the purpose of high-level activity recognition. To evaluate this work, 8 video sequences were captured in a controlled environment on a bus via two cameras, recording both the interior of the bus (covering 20 seats) and the door exterior. Actors were used to simulate passenger behaviour. Each sequence thus captures the activity of a number of passengers as they board the bus, move around the interior, sit down, stand up, and exit the bus. The vision analytics include classifiers for gender and orientation (sitting/standing), and a spatial location tracker. The uncertainty associated with detected events was modelled using Dempster-Shafer theory [26], similar to the event model in [16]. To evaluate our own framework, the authors provided us with the set of detected events for Sequence 1, as described in Appendix C of [11]. This video sequence involves two passengers who board the bus, sit down at a series of different seats, and then exit the bus. Although these events are referred to as composite events in their work, they represent primitive event observations in our case study.

The original data set and a complete SEC ProbLog implementation are available online,[4] along with full details about how the data set was translated. It suffices to say that translating the data set results in a probabilistic event narrative containing 62 probabilistic events, the first of which occurs at timepoint 53 and the last of which occurs at timepoint 2440. Each probabilistic event is comprised of a deterministic event type occurrence and a (ground) subset of the following probabilistic event type attributes:

subject(E, S). act(E, board). p :: act(E, exit).

$p_1 ::$ *gender(E, male)*; $p_2 ::$ *gender(E, female).*

$p_1 ::$ *location(E, door)*; $p_2 ::$ *location(E, gangway)*;

$p_3 ::$ *location(E, seat(1))*; …; $p_{22} ::$ *location(E, seat(20)).*

$p_1 ::$ *orientation(E, sitting)*; $p_2 ::$ *orientation(E, standing).*

Thus, the subject attribute and boarding event are both deterministic, while all other attributes are probabilistic.

In order to reason about the probabilistic event narrative, we completed the SEC ProbLog program by adding 12

[4] https://github.com/kevinmcareavey/secproblog

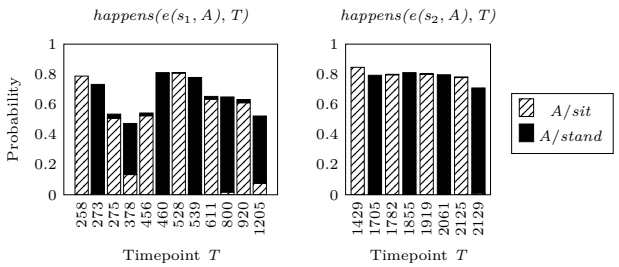

Figure 3: Probabilities of inferences in Section 5.

(deterministic) effect clauses and 2 probabilistic inference rules, along with axioms SEC1–3. Of the 12 effect clauses, 4 are taken directly from Example 2, with another 2 equivalent effect clauses used to initiate and terminate fluents of the form *location(S, L)*. The 6 remaining effect clauses describe the effect that boarding and exiting events have on the location and orientation fluents. In particular, boarding events are assumed to initiate the fluents *location(S, door)* and *orientation(S, standing)*, i.e. passengers board the bus at the door in standing orientation. Similarly, boarding and exiting events are assumed to terminate fluents of the form *location(S, L)* and *orientation(S, O)*, i.e. beliefs about a passenger's location and orientation are terminated when they exit the bus. For the inference rules, one was taken directly from Example 6, and the other was defined equivalently for stand events (and with the same probability).

Given the probabilistic event narrative, these effect clauses thus allow us to derive temporal probabilistic information about the gender, location, and orientation of each passenger for the duration of the sequence. The corresponding results for these fluents are equivalent to those already shown in Figure 2, so we do not repeat them here. For the two inference rules, results are summarised in Figure 3. From this figure we might conclude that passenger s_1 sits down at up to 6 different seats, while passenger s_2 sits down at up to 4 different seats. Note that there is no overlap between the sequence of inferred events associated with each passenger, although the data set suggests that both passengers are onboard between timepoints 1170 and 1395. While these results are fairly trivial (due to the limited data set), they serve to illustrate the ease with which SEC ProbLog programs can be used to formally reason about complex spatio-temporal information under uncertainty.

6. RELATED WORK

The closest work to ours is of course that of [29]. The relationship between the two frameworks can be stated as follows: the model in [29] is limited to the special case of Definition 1 where $n = 1$ and $p_1 \leq 1$. As such, their approach is effectively subsumed by ours. In comparison, our framework enhances expressivity and thus has broader applicability (e.g. [29] is not sufficiently expressive to model the real data set from Section 5). In Propositions 1 and 2, we also prove that the semantics of our approach are well-defined (i.e. valid in ProbLog) and this was not done in [29].

Although we have already mentioned the relative merits of the event calculus for event reasoning applications [19], there has also been some work in the broad area of PLPs on supporting uncertainty in similar frameworks. For example, hybrid PLPs with answer set semantics were used for probabilistic planning in [23]. Similarly, a probabilistic variant of action languages (which are closely related to LP) was proposed in [3]. As far as we are aware, no such approach uses annotated disjunctions or a similar PLP language construct. There has also been work in PLP on the more general problem of modelling time and dynamics. For example, dynamic variants of the PLP languages Causal Probabilistic Logic (CP-logic) [34] and Distributional Clauses (DCs) [10] were proposed in [32] and [21], respectively. Although these works do not relate directly to event reasoning, they may offer computational benefits for our extension of the SEC due to their specialised treatment of time.

7. CONCLUSION

Compared to the state-of-the-art literature on the event calculus, our extension is the first able to reason about uncertain event observations, uncertain effects, and uncertain composite event definitions. It is also the first able to reason about event observations from multiple sources. In addition to the case study in Section 5, we can find many suitable applications in the literature. For example, some uncertain event models from the literature can be trivially formulated as SEC ProbLog programs, such as [36] and [16][5]. If these existing frameworks rely on ad-hoc approaches to temporal reasoning (as is the case in [36, 16]), then SEC ProbLog programs offer a viable means to formalise their reasoning capabilities in a general way.

As mentioned previously, it is likely that our framework can generalise to at least some other PLPs. For example, the annotated disjunction plays a pivotal role in our framework definition, and this language construct is supported by at least one other PLP, namely Logic Programs with Annotated Disjunctions (LPADs) [35]. However, it is known [7] that other PLPs support similar language constructs. For example, the probabilistic clause from Stochastic Logic Programs (SLPs) [20], the probabilistic alternative from Independent Choice Logic (ICL) [22], the multi-ary random switch from Programming in Statistical Modelling (PRISM) [25], the CP-event from CP-logic [34], and the distributional clause from DCs [10]. For future work, we will investigate whether such a generalisation is possible, and if any PLP language is particularly suited to our framework.

It is worth noting that the ProbLog implementation of our framework operates in an offline fashion. However, there has already been some work on adapting the (classical) event calculus for real-time environments, e.g. in the cached event calculus [5], the reactive event calculus [4], and the runtime event calculus [2]. For future work, we also plan to study how our framework can be adapted for real-time environments in a similar way.

Acknowledgments

This work has received funding from the EPSRC PACES project (EP/J012149/1) and the European Union's Horizon 2020 research and innovation programme through the DEVELOP project, under grant agreement No. 688127. The authors would like to thank Xin Hong for providing the event detection data set discussed in Section 5.

[5]Assuming the events' mass functions are restricted to Bayesian mass functions [26].

REFERENCES

[1] J. F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–154, 1984.

[2] A. Artikis, M. Sergot, and G. Paliouras. An event calculus for event recognition. *IEEE Transactions on Knowledge and Data Engineering*, 27(4):895–908, 2015.

[3] C. Baral, N. Tran, and L.-C. Tuan. Reasoning about actions in a probabilistic setting. In *Proc. of 18th Nat. Conf. on Artificial Intelligence*, pages 507–512, 2002.

[4] F. Chesani, P. Mello, M. Montali, and P. Torroni. A logic-based, reactive calculus of events. *Fundamenta Informaticae*, 105(1-2):135–161, 2010.

[5] L. Chittaro and A. Montanari. Efficient temporal reasoning in the cached event calculus. *Computational Intelligence*, 12(3):359–382, 1996.

[6] K. L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 293–322. Springer, 1978.

[7] L. De Raedt and A. Kimmig. Probabilistic (logic) programming concepts. *Machine Learning*, 100(1):5–47, 2015.

[8] D. Fierens, G. Van den Broeck, J. Renkens, D. Shterionov, B. Gutmann, I. Thon, G. Janssens, and L. De Raedt. Inference and learning in probabilistic logic programs using weighted Boolean formulas. *Theory and Practice of Logic Programming*, 15(3):358–401, 2015.

[9] M. Gelfond and V. Lifschitz. Representing action and change by logic programs. *The Journal of Logic Programming*, 17(2):301–321, 1993.

[10] B. Gutmann, I. Thon, A. Kimmig, M. Bruynooghe, and L. De Raedt. The magic of logical inference in probabilistic programming. *Theory and Practice of Logic Programming*, 11(4-5):663–680, 2011.

[11] X. Hong, Y. Huang, W. Ma, S. Varadarajan, P. Miller, W. Liu, M. Jose Santofimia Romero, J. Martinez del Rincon, and H. Zhou. Evidential event inference in transport video surveillance. *Computer Vision and Image Understanding*, 144:276–297, 2016.

[12] R. Kowalski. Database updates in the event calculus. *The Journal of Logic Programming*, 12(1-2):121–146, 1992.

[13] R. Kowalski and M. Sergot. A logic-based calculus of events. *New Generation Computing*, 4(1):67–95, 1986.

[14] V. Lifschitz. Twelve definitions of a stable model. In *Proc. of 24th Int. Conf. on Logic Programming*, pages 37–51, 2008.

[15] J. Ma, W. Liu, and P. Miller. Handling sequential observations in intelligent surveillance. In *Proc. of 5th Int. Conf. on Scalable Uncertainty Management*, pages 547–560, 2011.

[16] J. Ma, W. Liu, P. Miller, and W. Yan. Event composition with imperfect information for bus surveillance. In *Proc. of 6th IEEE Int. Conf. on Advanced Video and Signal Based Surveillance*, pages 382–387, 2009.

[17] J. McCarthy. Circumscription—a form of non-monotonic reasoning. *Artificial Intelligence*, 13(1-2):27–39, 1980.

[18] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of Artificial Intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 4, pages 463–502. Edinburgh University Press, 1969.

[19] E. T. Mueller. *Commonsense Reasoning: An Event Calculus Based Approach*. Morgan Kaufmann, second edition, 2015.

[20] S. Muggleton. Stochastic logic programs. *Advances in Inductive Logic Programming*, 32:254–264, 1996.

[21] D. Nitti, T. D. Laet, and L. De Raedt. A particle filter for hybrid relational domains. In *Proc. of 2013 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 2764–2771, 2013.

[22] D. Poole. The independent choice logic for modelling multiple agents under uncertainty. *Artificial Intelligence*, 94(1):7–56, 1997.

[23] E. Saad. Probabilistic planning in hybrid probabilistic logic programs. In *Proc. of 1st Int. Conf. on Scalable Uncertainty Management*, pages 1–15, 2007.

[24] T. Sato. A statistical learning method for logic programs with distributional semantics. In *Proc. of 12th Int. Conf. on Logic Programming*, pages 715–729, 1995.

[25] T. Sato and Y. Kameya. Parameter learning of logic programs for symbolic-statistical modeling. *Journal of Artificial Intelligence Research*, 15:391–454, 2001.

[26] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

[27] M. Shanahan. *Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia*. The MIT Press, 1997.

[28] A. Skarlatidis. *Event Recognition Under Uncertainty and Incomplete Data*. PhD thesis, University of Piraeus, 2014.

[29] A. Skarlatidis, A. Artikis, J. Filippou, and G. Paliouras. A probabilistic logic programming event calculus. *Theory and Practice of Logic Programming*, 15(2):213–245, 2015.

[30] A. Skarlatidis, G. Paliouras, A. Artikis, and G. A. Vouros. Probabilistic event calculus for event recognition. *ACM Transactions on Computational Logic*, 16(2):11:1–11:37, 2015.

[31] M. Stone. The opinion pool. *The Annals of Mathematical Statistics*, 32(4):1339–1342, 1961.

[32] I. Thon, N. Landwehr, and L. De Raedt. Stochastic relational processes: Efficient inference and applications. *Machine Learning*, 82(2):239–272, 2011.

[33] A. Van Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):619–649, 1991.

[34] J. Vennekens. *Algebraic and logical study of constructive processes in knowledge representation*. PhD thesis, K.U. Leuven, 2007.

[35] J. Vennekens, S. Verbaeten, and M. Bruynooghe. Logic programs with annotated disjunctions. In *Proc. of 20th Int. Conf. on Logic Programming*, pages 431–445, 2004.

[36] S. Wasserkrug and O. Etzion. A model for reasoning with uncertain rules in event composition systems. In *Proc. of 21st Conf. on Uncertainty in Artificial Intelligence*, pages 599–608, 2005.